

## PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2001-101045

(43)Date of publication of application : 13.04.2001

(51)Int.Cl.

G06F 12/00

(21)Application number : 11-276264

(71)Applicant : TOSHIBA CORP

(22)Date of filing : 29.09.1999

(72)Inventor : KANAI TATSUNORI

TORII OSAMU

KITSU TOSHIKI

MAEDA SEIJI

YAO HIROSHI

YANO HIROKUNI

## (54) METHOD AND SYSTEM FOR PROCESSING TRANSACTION

## (57)Abstract:

PROBLEM TO BE SOLVED: To provide a transaction processing system for easily and efficiently realizing file update processing and recovery processing on the occurrence of any failure by plural transactions operating in parallel.

SOLUTION: As for a file in which UPDATE or REMOVE or the like is updated by plural transactions (TXID=1-3) operating in parallel, when one transaction (TXID=1) which updates this file commits, information (TXID=2 and TXSTATE=UPDATE, <oldvalue> 200 yen </oldvalue> or the like in <entry>) necessary for the update of the file by the transaction but also the update of the file by the other transactions (TXID=2, 3) and the cancellation of this is written in the file in a batch. Then, recovery processing after the occurrence of any failure is operated at the point of using the file the next.

```

<KVtable>
  <entry TXID="5" TXSTATE="UPDATE">
    <key>APPLE</key>
    <oldvalue>500Y</oldvalue>
    <value>400Y</value>
  </entry>
  <entry TXID="2" TXSTATE="REMOVE">
    <key>BISCUIT</key>
    <value>250Y</value>
  </entry>
  <entry TXID="2" TXSTATE="UPDATE">
    <key>CHOCOLATE</key>
    <oldvalue>300Y</oldvalue>
    <value>400Y</value>
  </entry>
  <entry TXID="1" TXSTATE="REMOVE">
    <key>GRAPE</key>
    <value>500Y</value>
  </entry>
  <entry TXID="1" TXSTATE="UPDATE">
    <key>ICECREAM</key>
    <value>450Y</value>
  </entry>
  <entry TXID="2" TXSTATE="UPDATE">
    <key>KIWIFRUIT</key>
    <value>300Y</value>
  </entry>
</KVtable>

```

## LEGAL STATUS

[Date of request for examination]

01.12.2004

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開2001-101045

(P2001-101045A)

(43) 公開日 平成13年4月13日 (2001. 4. 13)

(51) Int. Cl.<sup>7</sup>

G 0 6 F 12/00

識別記号

5 1 8

F I

G 0 6 F 12/00

テームコード (参考)

5 1 8 A 5 B 0 8 2

審査請求 未請求 請求項の数 8 O L (全 26 頁)

(21) 出願番号 特願平11-276264

(22) 出願日 平成11年9月29日 (1999. 9. 29)

(71) 出願人 000003078

株式会社東芝

神奈川県川崎市幸区堀川町72番地

(72) 発明者 金井 達徳

神奈川県川崎市幸区小向東芝町1番地 株式会社東芝研究開発センター内

(72) 発明者 鳥井 修

神奈川県川崎市幸区小向東芝町1番地 株式会社東芝研究開発センター内

(74) 代理人 100058479

弁理士 鈴江 武彦 (外6名)

最終頁に続く

(54) 【発明の名称】 トランザクション処理方法及びトランザクション処理システム

(57) 【要約】

【課題】 並行に動作する複数のトランザクションによるファイルの更新処理と障害時のリカバリ処理を簡単に効率良く実現することのできるトランザクション処理システムを提供すること。

【解決手段】 並行に動作する複数のトランザクション (TXID=1~3) によってUPDATEやREMOVEなどの更新をされたファイルに対して、それを更新した1つのトランザクション (TXID=1) のコミット時に、そのトランザクションによる更新だけでなく、他のトランザクション (TXID=2, 3) による更新とそれを取り消すのに必要な情報 (<entry>におけるTXID=2とTXSTATE=UPDATE、<oldvalue>200円</oldvalue>等) を、同じファイルにまとめて書き込む。障害発生後のリカバリ処理は次にそのファイルを使う時点で行えばよい。

```
<KVtable>
<entry TXID="3" TXSTATE="UPDATE">
  <key>APPLE</key>
  <value>500円</value>
  <oldvalue>400円</oldvalue>
</entry>
<entry TXID="2" TXSTATE="REMOVE">
  <key>BISCUIT</key>
  <value>250円</value>
</entry>
<entry TXID="2" TXSTATE="UPDATE">
  <key>CHOCOLATE</key>
  <value>300円</value>
  <oldvalue>200円</oldvalue>
</entry>
<entry TXID="1" TXSTATE="REMOVE">
  <key>GRAPE</key>
  <value>800円</value>
</entry>
<entry TXID="1" TXSTATE="UPDATE">
  <key>ICECREAM</key>
  <value>450円</value>
</entry>
<entry TXID="2" TXSTATE="UPDATE">
  <key>KIWIFRUIT</key>
  <value>300円</value>
</entry>
</KVtable>
```

1

## 【特許請求の範囲】

【請求項1】 バッファ領域上に読み出した同一のファイルを更新対象として複数のトランザクションが処理を行い、

前記複数のトランザクションのうちの一つのトランザクションのコミット時に、該一つのコミットするトランザクションによる確定した更新内容と、他のコミットしていないトランザクションによる未確定の更新内容と、該未確定の更新を取り消すための情報とをファイル内容として含む前記ファイルを、安定記憶装置に書き込むことを特徴とするトランザクション処理方法。

【請求項2】 前記他のコミットしていないトランザクションをコミットする時には、該他のこれからコミットするトランザクションによる新たに確定した更新内容と、以前にコミットしたトランザクションによる既に確定している更新内容のうち該他のこれからコミットするトランザクションによる新たに確定した更新内容とは競合しないものとをファイル内容として含むファイルを、前記安定記憶装置に格納された前記ファイルに上書きすることを特徴とする請求項1に記載のトランザクション処理方法。

【請求項3】 新しくファイルを前記安定記憶装置から読み出した時に、このファイルに、コミットしていないトランザクションによる未確定の更新内容および該未確定の更新を取り消すための情報が含まれているか否かを調べ、含まれている場合には、読み出した前記ファイルを前記情報を用いて前記コミットしていないトランザクションによる更新がなかった状態に戻すことを特徴とする請求項1または2に記載のトランザクション処理方法。

【請求項4】 前記一つのコミットするトランザクションが前記ファイルとは別のファイルをも更新対象として前記バッファ領域上で処理を行っていた場合には、前記ファイルの前記安定記憶装置への書き込みと同期させて、該一つのコミットするトランザクションによる前記別のファイルに対する確定した更新内容をファイル内容として含む前記別のファイルを前記安定記憶装置に書き込むことを特徴とする請求項1ないし3のいずれか1項に記載のトランザクション処理方法。

【請求項5】 前記ファイルは、一纏まりの情報を分割して記録した複数個のファイルの集合の中の一つのファイルであることを特徴とする請求項1ないし4のいずれか1項に記載のトランザクション処理方法。

【請求項6】 一纏まりの情報を分割して記録した複数個のファイルの集合に対して二つ以上のトランザクションが前記バッファ領域上で更新を行い、この二つ以上のトランザクションによる更新内容に基づいて、前記バッファ領域上で、新しいファイルの追加、ファイル間の情報の移動および不要ファイルの削除のうち少なくとも一つを行うことにより、前記複数個のファ

2

イルそれぞれの大きさが予め定められた基準に従うように調整し、

前記一つのコミットするトランザクションが更新したファイル、および該トランザクションによる更新に伴う前記調整の結果更新あるいは新規に作成されたファイルのうち、少なくとも一つのファイルに対して他のトランザクションが更新を行っている場合には、該少なくとも一つのファイルを更新対象として複数のトランザクションが処理を行ったものとして、前記ファイルの前記安定記憶装置への書き込みを行うことを特徴とする請求項1ないし4のいずれか1項に記載のトランザクション処理方法。

【請求項7】 トランザクションの処理対象となるファイルを安定記憶装置からバッファ領域上に読み出す手段と、

前記ファイルに対して複数のトランザクションのそれぞれが行った更新の内容と該更新を取り消すための情報とを前記バッファ領域上で該ファイル内に書き込む手段と、

前記複数のトランザクションのうちの一つのトランザクションのコミット時に、前記ファイル内に書き込まれた該一つのコミットするトランザクションによる更新の内容については該更新を取り消すための情報を削除することにより確定させ、前記ファイル内に書き込まれた他のコミットしていないトランザクションによる更新の内容および該更新を取り消すための情報についてはそのままとし、確定させた内容とそのままとした情報とをファイル内容として含む前記ファイルを、前記安定記憶装置に書き出す手段とを備えたことを特徴とするトランザクション処理システム。

【請求項8】 トランザクションの処理対象となる一纏まりの情報を分割して記録した複数個のファイルの集合を安定記憶装置からバッファ領域上に読み出す手段と、前記ファイルの集合に対して複数のトランザクションのそれぞれが行った更新の内容に基づいて、前記バッファ領域上で、該更新の内容と該更新を取り消すための情報とをファイル内に書き込むとともに、新しいファイルの追加、ファイル間の情報の移動および不要ファイルの削除のうち少なくとも一つを行うことにより、前記複数個のファイルそれぞれの大きさが予め定められた基準に従うように調整する手段と、

前記複数のトランザクションのうちの一つのトランザクションのコミット時に、前記ファイル内に書き込まれた該一つのコミットするトランザクションによる更新の内容については該更新を取り消すための情報を削除することにより確定させ、前記ファイルおよび前記調整の結果更新あるいは新規に作成されたファイルのうちの少なくとも一つのファイルに対して更新を行った他のコミットしていないトランザクションによる、該少なくとも一つのファイル内に書き込まれた更新の内容および該更新を

## 3

取り消すための情報についてはそのままとし、確定させた内容をファイル内容として含む前記ファイルとそのままとした情報をファイル内容として含む前記少なくとも一つのファイルとを、前記安定記憶装置に書き出す手段とを備えたことを特徴とするトランザクション処理システム。

## 【発明の詳細な説明】

## 【0001】

【発明の属する技術分野】本発明は、複数のトランザクションが並行して同じファイルにアクセスするトランザクション処理方法及びトランザクション処理システムに関する。

## 【0002】

【従来の技術】一般に、トランザクション処理システムにおいては、トランザクションと呼ぶ処理の流れを単位として処理の実行を管理する。例えば、端末装置やネットワークなどから受け取った処理要求に対して、必要な処理を実行し、その結果を返すような、何らかの手段で起動された処理を実行する流れをトランザクションと呼ぶ。個々のトランザクションは、実行過程で、1または複数のファイル（あるいはデータベース）に記録管理するデータの更新を行う。トランザクション処理システムは、このような1または複数のデータの更新を、アトミックにコミットするかアボートするかのいずれかの状態にすることを保証する。すなわち、すべての更新を同時に有効にするのがコミットである。逆に、すべての更新を捨ててトランザクションを実行しなかったのと同じ状態に戻るのがアボートである。また、トランザクション処理システムは、一度トランザクションをコミットしたら、そのトランザクションによる更新が取り消されることがないことも保証する。

【0003】このようなトランザクション処理システムを実現する方式は広く知られており、例えば“TRANSACTION PROCESSING: CONCEPTS AND TECHNIQUES”（Jim Gray, Andreas Reuter 著, Morgan Kaufmann）にはさまざまな方式が開示されている。

【0004】トランザクション処理システムにおいて扱いが難しいのは、1つのファイル上のデータを、同時に複数のトランザクションが更新する場合である。1つのファイルには同時に1つのトランザクションしか更新することを許さなければ問題は生じない。しかし、同時に複数のトランザクションを並行に処理して性能を向上させるためには、同時に1つのファイルの中の異なる部分に記録管理するデータを更新できるようにすることが必要になる。この問題は、ファイルではなくデータベースを用いる場合も同様である。以下、ファイルを例に説明するが、データベースにおいても同様である。

【0005】このような並行に更新処理を行うトランザ

## 4

クション処理システムを実現するために従来から広く用いられているのは、WAL（Write Ahead Logging）と呼ばれる方式である。この方式は、メモリ上のバッファに読み込んだファイルに対し、複数のトランザクションが更新を行った場合、ログファイルにはどのトランザクションがどのファイルのどの部分をどう書き換えたかを記録する。さらに、ログに記録した更新情報が安定記憶装置（例えばハードディスク）上に書かれて消えないことが保証されるまで、バッファ上で行った更新はハードディスクなどの記憶媒体上の元のファイルには書き戻さないようにタイミングを制御する。

【0006】WAL方式ではすべての更新履歴はログに記録されており、システムの障害が発生した場合には、そのログを使ってファイルを正しい状態に戻すことができる。すなわち、障害が発生してシステムを再起動した場合、トランザクション処理システムはログを見て、コミットしたトランザクションが行った更新で、それがファイルに反映されていないものは正しく反映させる。また、アボートしたトランザクションによる更新がファイルに書きこまれていたらそれを取り消す処理を行う。このようなリカバリ処理により、障害発生に対してもトランザクションの原子性（Atomicity）や耐久性（Durability）や分離性（Isolation）や一貫性（Consistency）を保証することができる。

【0007】トランザクション処理システムを実現するもう1つの方式は、シャドウページ方式である。この方式でファイルを更新する場合には、ハードディスク上に、更新前のファイルが書き込まれているページと、更新後のファイルが書き込まれているページの両方を用意し、トランザクションのコミット時にファイルの管理データをアトミックに入れ替える。そのため、常にハードディスク上には正しいファイルが存在することが保証できる。

## 【0008】

【発明が解決しようとする課題】従来のWAL方式によるトランザクション処理方式では、トランザクションがファイルを更新する場合、まず更新内容をログに書いてからファイル本体を更新するため、複数のトランザクションによるファイルの共有を管理するためには複雑な処理手順を必要とする。また、1回の更新に対して、ログとファイル本体への2回の書き込み処理が必要になる。また、リカバリに必要な更新情報をログに記録し、リカバリ時にそのログを使ってファイルを一貫性のある状態に戻す処理は、複雑な処理手順を必要とする。さらに、リカバリ処理は障害発生後の再起動時に集中して行わなければならない、再起動に時間がかかるという問題点がある。

【0009】一方、シャドウページ方式では、ハードディスク上には常に正しい状態のファイルが存在するの

5

で、障害発生時のリカバリ処理のオーバーヘッドが小さいという特徴を持つ。しかし、シャドウページ方式は、1つのトランザクションのコミット前とコミット後の2つのファイルのデータが記録されたページを、アトミックに交換することを基本操作としている。そのため、複数のトランザクションが同じファイルを更新したい場合には、それらのトランザクションをシリアライズして順に更新して行くようにしなければならない。WAL方式のように同じファイルの異なる部分を複数のトランザクションが並行して更新できるようにするためにはシャドウページ方式だけでは無理で、その上にWAL方式と同じようにログを使ったデータの管理をするなどの工夫が必要になる。そのため、シャドウページ方式は簡単に実装できても、その上に複雑なデータの管理を実装することが必要になるという問題点がある。

【0010】本発明は、上記事情を考慮してなされたもので、並行に動作する複数のトランザクションによるファイルの更新処理と障害時のリカバリ処理を簡単に効率良く実現することのできるトランザクション処理方法及びトランザクション処理システムを提供することを目的とする。

【0011】

【課題を解決するための手段】本発明（請求項1）に係るトランザクション処理方法は、バッファ領域上に読み出した同一のファイルを更新対象として複数のトランザクションが処理を行い、前記複数のトランザクションのうちの一つのトランザクションのコミット時に、該一つのコミットするトランザクションによる確定した更新内容と、他のコミットしていないトランザクションによる未確定の更新内容と、該未確定の更新を取り消すための情報とをファイル内容として含む前記ファイルを、安定記憶装置に書き込むことを特徴とする。

【0012】安定記憶装置とは、例えば、ハードディスク等の、電源を切っても記憶した内容が消えない記憶装置を指す。

【0013】本発明では、上記のように、あるトランザクションのコミット時に、そのトランザクションが更新したファイルを別のトランザクションも更新している場合には、その別のトランザクションによる更新内容とそれを取り消すための情報も一緒にファイルの中身へ書き込んで、そのファイルを安定記憶装置（例えばハードディスク）へ書き出すことにより、ファイルとは別にログを用意して複雑な管理を行うようなことをしなくとも、トランザクションの原子性（トランザクション単位で、行った更新の全てが安定記憶装置上で有効になる（コミット）か全て無効になる（アボート）かいずれかの状態にしかならないことを、システムに障害が起きた場合でも保証できること）を担保しつつ、一つのファイルのデータを複数のトランザクションが並行して更新することを実現できる。

6

【0014】好ましくは、前記他のコミットしていないトランザクションをコミットする時には、該他のこれからコミットするトランザクションによる新たに確定した更新内容と、以前にコミットしたトランザクションによる既に確定している更新内容のうち該他のこれからコミットするトランザクションによる新たに確定した更新内容とは競合しないものとをファイル内容として含むファイルを、前記安定記憶装置に格納された前記ファイルに上書きするようにしてもよい。

10 【0015】あるトランザクションがコミットした後、同じファイルを更新している別のトランザクションをコミットする場合には、後にコミットするトランザクションの更新内容と、先にコミットしたトランザクションの更新内容のうち後にコミットするトランザクションによって書き換えられてはいないものとを書き込んだファイルで、先に安定記憶装置に書き出されたファイル（後にコミットすることになるトランザクションの更新内容とそれを取り消すための情報が入っている）を上書きすれば、トランザクションの原子性を担保できる。なお、あるトランザクションがコミットした後、同じファイルを更新している別のトランザクションをアボートする場合には、アボートするトランザクションによる更新がなかった状態に戻したファイルで、先に安定記憶装置に書き出されたファイル（後にアボートすることになるトランザクションの更新内容とそれを取り消すための情報が入っている）を上書きしても良いし、アボート時には何もしなくとも、下記のように次にファイルを読み出す時にリカバリ処理を行えば、トランザクションの原子性は担保できる。

30 【0016】好ましくは、新しくファイルを前記安定記憶装置から読み出した時に、このファイルに、コミットしていないトランザクションによる未確定の更新内容および該未確定の更新を取り消すための情報が含まれているか否かを調べ、含まれている場合には、読み出した前記ファイルを前記情報を用いて前記コミットしていないトランザクションによる更新がなかった状態に戻すようにしてもよい。

40 【0017】あるトランザクションがコミットした後、同じファイルを更新している別のトランザクションのコミット／アボートが行われる前に、システムに障害が起きた場合でも、コミットしたトランザクションによる更新は既に安定記憶装置上でファイルに正しく反映されている。これに加えて上記のようなりカバリ処理を行えば、未確定の（別の）トランザクションによる更新はアボートしたものとしてそれを取り消すことができるので、トランザクションの一貫性が保たれる。しかも、このリカバリ処理は、障害回復時に行う必要はなく、次にそのファイルを読み出した時に行えば良い。

50 【0018】好ましくは、前記一つのコミットするトランザクションが前記ファイルとは別のファイルをも更新

7

対象として前記バッファ領域上で処理を行っていた場合には、前記ファイルの前記安定記憶装置への書き込みと同期させて、該一つのコミットするトランザクションによる前記別のファイルに対する確定した更新内容をファイル内容として含む前記別のファイルを前記安定記憶装置に書き込むようにしてもよい。

【0019】上記のように複数のファイルの書き込みを同期させれば、一つのトランザクションがその処理開始時からコミット時まで複数のファイルに対して行った更新の全てが安定記憶装置上で有効になる（コミット）か、全てが無効になる（アボート）か、いずれかの状態にしかならない（システムに障害が起きた場合でも、一つのトランザクションが行った更新のうち一部分だけが有効になり残りが無効になるということはある得ない）ことを保証するような、アトミックな書き込みが実現できる。

【0020】好ましくは、前記ファイルは、一纏まりの情報を分割して記録した複数のファイルの集合の中の一つのファイルであるようにしてもよい。

【0021】好ましくは、一纏まりの情報を分割して記録した複数のファイルの集合に対して二つ以上のトランザクションが前記バッファ領域上で更新を行い、この二つ以上のトランザクションによる更新内容に基づいて、前記バッファ領域上で、新しいファイルの追加、ファイル間の情報の移動および不要ファイルの削除のうち少なくとも一つを行うことにより、前記複数のファイルそれぞれの大きさが予め定められた基準に従うように調整し、前記一つのコミットするトランザクションが更新したファイル、および該トランザクションによる更新に伴う前記調整の結果更新あるいは新規に作成されたファイルのうち、少なくとも一つのファイルに対して他のトランザクションが更新を行っている場合には、該少なくとも一つのファイルを更新対象（すなわち前記ファイル）として複数のトランザクションが処理を行ったものとして、前記ファイルの前記安定記憶装置への書き込みを行うようにしてもよい。

【0022】また、本発明（請求項7）に係るトランザクション処理システムは、トランザクションの処理対象となるファイルを安定記憶装置からバッファ領域上に読み出す手段と、前記ファイルに対して複数のトランザクションのそれぞれが行った更新の内容と該更新を取り消すための情報とを前記バッファ領域上で該ファイル内に書き込む手段と、前記複数のトランザクションのうちの一つのトランザクションのコミット時に、前記ファイル内に書き込まれた該一つのコミットするトランザクションによる更新の内容については該更新を取り消すための情報を削除することにより確定させ、前記ファイル内に書き込まれた他のコミットしていないトランザクションによる更新の内容および該更新を取り消すための情報についてはそのままとし、確定させた内容とそのままとし

8

た情報とをファイル内容として含む前記ファイルを、前記安定記憶装置に書き出す手段とを備えたことを特徴とする。

【0023】また、本発明（請求項8）に係るトランザクション処理システムは、トランザクションの処理対象となる一纏まりの情報を分割して記録した複数のファイルの集合を安定記憶装置からバッファ領域上に読み出す手段と、前記ファイルの集合に対して複数のトランザクションのそれぞれが行った更新の内容に基づいて、前記バッファ領域上で、該更新の内容と該更新を取り消すための情報とをファイル内に書き込むとともに、新しいファイルの追加、ファイル間の情報の移動および不要ファイルの削除のうち少なくとも一つを行うことにより、前記複数のファイルそれぞれの大きさが予め定められた基準に従うように調整する手段と、前記複数のトランザクションのうちの一つのトランザクションのコミット時に、前記ファイル内に書き込まれた該一つのコミットするトランザクションによる更新の内容については該更新を取り消すための情報を削除することにより確定させ、前記ファイルおよび前記調整の結果更新あるいは新規に作成されたファイルのうち少なくとも一つのファイルに対して更新を行った他のコミットしていないトランザクションによる、該少なくとも一つのファイル内に書き込まれた更新の内容および該更新を取り消すための情報についてはそのままとし、確定させた内容をファイル内容として含む前記ファイルとそのままとした情報をファイル内容として含む前記少なくとも一つのファイルとを、前記安定記憶装置に書き出す手段とを備えたことを特徴とする。

【0024】なお、装置に係る本発明は方法に係る発明としても成立し、方法に係る本発明は装置に係る発明としても成立する。

【0025】また、装置または方法に係る本発明は、コンピュータに当該発明に相当する手順を実行させるための（あるいはコンピュータを当該発明に相当する手段として機能させるための、あるいはコンピュータに当該発明に相当する機能を実現させるための）プログラムを記録したコンピュータ読取り可能な記録媒体としても成立する。

【0026】本発明によれば、並行に動作する複数のトランザクションによるファイルの更新を、簡単な処理手順で実現することができる。本発明によれば、従来のWAL方式や、シャドウページ方式の上に特殊なログ管理を実装するときのように、データの更新時やコミット処理時あるいはリカバリ時に複雑な処理を必要としない。

【0027】また、本発明では、障害発生後のリカバリ処理を、再起動時に集中して行う必要は無く、障害で更新途中の状態は次にそのファイルを使う時点で行われるため、リカバリ処理のオーバーヘッドが小さい。

【0028】さらに、本発明では、1つのデータを複数

のファイルに分けて管理することで、ファイルの更新のコミット時にディスクに書き戻すデータの量を減らすことができる。

【0029】

【発明の実施の形態】以下、図面を参照しながら発明の実施の形態を説明する。

【0030】最初に従来のファイルシステムについて簡単に説明する。図43に、従来のトランザクション処理システムの例を示す。従来のトランザクション処理システムでは、トランザクションを処理するアプリケーションプログラム101は、データベース管理システム102を介してデータの操作を行う。ファイルシステム103はハードディスク104などの安定記憶装置に記録したファイルの操作や管理を行う。データベース管理システム102は、ファイルシステム103の管理するファイルを使って、アプリケーションプログラム101が必要とするデータを管理する。複数のアプリケーションプログラム101が共有しているデータに対する操作の間の並行制御もデータベース管理システム102が行う。

【0031】図1に、本発明の一実施形態に係るトランザクション処理システムの構成例を示す。

【0032】なお、本実施形態では、安定記憶装置としてハードディスクを用いるものとして説明するが、もちろん安定記憶装置として他の装置を使用することも可能である。また、本実施形態では、ユーザプログラムによりトランザクションが発生するものとして説明するが、もちろんユーザプログラム以外のものがトランザクションを発生させる場合も同様である。

【0033】本実施形態のトランザクション処理システムでは、図1に示されるように、トランザクションを実行する複数のアプリケーションプログラム2間で共有するファイルの操作は、共有データ管理部4を介して行い、共有しないファイルの操作は、アプリケーションプログラム2がファイルシステム6を直接指示して行う。

【0034】トランザクションマネージャ8は、各アプリケーションプログラム2が処理するトランザクションの管理を行う。アプリケーションプログラム2は、トランザクションの処理を開始する時点で、トランザクションマネージャ8からトランザクション識別子を割り付けもらう。以降、トランザクションの処理を進めて行く過程で、共有データ管理部4やファイルシステム6に操作を指示するときには、トランザクション識別子を提示してどのトランザクションによるファイルの更新であるかを知らせる。アプリケーションプログラム2がトランザクションの処理の終了をトランザクションマネージャ8に伝えたと、トランザクションマネージャ8は共有データ管理部4やファイルシステム6に対して、トランザクション識別子で指定したトランザクションによるファイルの更新結果をハードディスク10に書き込んでコミットするか、あるいは更新結果を廃棄してアボートする

かを指示する。

【0035】なお、トランザクションマネージャ8を用いず、ユーザプログラム2が自分でトランザクション識別子に相当する情報を生成したり、コミット（COMMIT）やアボート（ABORT）を指示するように実施することもできる（本実施形態では、トランザクションマネージャ8を用いる場合について説明する）。

【0036】共有データ管理部4は、複数のトランザクションによる同一ファイルの異なる部分の更新を実現する。共有データ管理部4は、（当該共有データ管理部4を介して）どのファイルがどのトランザクションによって更新されているかを、更新管理表41を用いて管理している。図2に、更新管理表41の例を示す。更新管理表41には、ファイル名と、そのファイルを更新中のトランザクションのトランザクション識別子のリストの対応が記録されている。図2では、例えば、F0102というファイルはトランザクション識別子が36、21、5の3つのトランザクションによって更新中であることを示している。

【0037】ファイルシステム6は、トランザクションによるファイルの操作を実現する部分である。ファイルシステム6は、バッファ領域62を管理するバッファ管理部（図示せず）とトランザクション管理表61を持っている。ハードディスク10に記録しているファイルを読んだり更新したりする場合には、そのファイルをバッファ領域62に読み出してきて、バッファ領域62上で読んだり更新したりする。新しくファイルを作成する場合には、まずバッファ領域62にファイルを作成する。バッファ領域62上で更新や作成されたファイルは、トランザクションのコミット時にハードディスク10に書き込む。トランザクション管理表61は、どのトランザクションがどのファイルを更新しているかを管理している。

【0038】図3に、トランザクション管理表61の例を示す。トランザクション管理表61には、トランザクション識別子と、そのトランザクションが更新中のファイルのリストの対応が記録されている。図3の例では、トランザクション識別子“3”のトランザクションはファイルF9268のみを更新しており、トランザクション識別子“5”のトランザクションはファイルF0100とF9264の2つのファイルを更新していることを示している。トランザクション管理表61の最後のエンタリは、どのトランザクションにも属していないファイルの更新を管理している。

【0039】図3に例示したトランザクション管理表61では、最後の行のトランザクション“NOTX”のエンタリがこれに当たる。この例では、ファイルF0102、F2156、F3624、F5497、F7531は特定のトランザクションの一部として更新されているのではなく、どのトランザクションとも関連せずに更新

11

されていることを示す。以下で説明するように、共有データ管理部4を介して更新中のファイルは、どのトランザクションとも関連しない状態で更新するので、このエントリに記録される。なお、このエントリに記録されるファイルがどのトランザクションによって更新されているかは、共有データ管理部4の更新管理表41を参照すれば知ることができる。

【0040】以下は、データの一例としてXMLドキュメントの形式での情報を例にとり、共有データ管理部4を介して、このXMLドキュメントの形式で情報が記録されているファイルを更新するトランザクション処理システムに本発明を適応した場合についてより具体的に説明する。

【0041】なお、XMLに関しては、“Extensible Markup Language (XML) 1.0” (W3C Recommendation 10-Feb-98) に開示されている。

【0042】この場合、トランザクション処理システムでは、「キー」とそれに対する「値」のペアの集合を、XMLのドキュメントの形式でファイルに記録して管理している。このドキュメントは、キーを指定してその値を検索するという、一般的なデータベースとして使用することができる。

【0043】図4に、そのようなドキュメントの一例を示す。この例のXMLドキュメントでは、<key>と</key>のタグで囲まれた部分に「キー」を保持し、<value>と</value>のタグで囲まれた部分に「値」を保持し、そのペアを<entry>と</entry>のタグで囲んだエントリを作り、それらが複数集まったものを、最も外の<KVtable>と</KVtable>のタグが囲んでいる。実際には、XMLドキュメントは先頭部分に<?xml>で始まるプロローグ部を持つが、ここでは省略する。なお、XMLの用語として、<key>のような開始タグと、</key>のようなそれに対応する終了タグで囲まれた部分を、エレメントと呼ぶ。

【0044】図4に例示したXMLドキュメントでは、キー「APPLE」に対応する値は「400円」であり、キー「BISCUIT」に対応する値は「250円」であり、キー「CHOCOLATE」に対応する値は「200円」であり、キー「GRAPE」に対応する値は「600円」になっている。

【0045】このXMLドキュメントのファイルに対して、トランザクションを実行するアプリケーションプログラム2およびトランザクションマネージャ8は、共有データ管理部4を介して、READ (txid, key

12

\*y)とUPDATE (txid, key, value)とREMOVE (txid, key)とCOMMIT (txid)とABORT (txid)の5種類の操作ができるものとする。本実施形態では、READ、UPDATE、REMOVEはアプリケーションプログラム2が指示し、COMMITとABORTはトランザクションマネージャ8が指示するものとする。なお、前述したように、トランザクションマネージャ8が存在せず、アプリケーションプログラム2がCOMMITやABORTも指示するように実施することもできる。

【0046】READ (txid, key)は、txidで指定したトランザクション識別子を持つトランザクションが、keyで指定したエントリの値を読み出す操作である。

【0047】UPDATE (txid, key, value)は、txidで指定したトランザクション識別子を持つトランザクションが、keyで指定したエントリの値をvalueで指定した値に更新する操作である。keyで指定したエントリが無い場合には、新しくエントリを作成する。

【0048】REMOVE (txid, key)は、txidで指定したトランザクション識別子を持つトランザクションが、keyで指定したエントリを削除する操作である。

【0049】COMMIT (txid)は、txidで指定したトランザクション識別子を持つトランザクションが行ったファイルの内容の変更を、ディスク上のファイルに書き戻して、変更を永久的なものにする。

【0050】ABORT (txid)は、txidで指定したトランザクション識別子を持つトランザクションが行ったファイルの内容の変更を、すべて取り消して無かったものにする。

【0051】これらの5つの操作のうち、READはファイルの内容を変更しない操作であるのに対し、UPDATEとREMOVEはファイルの内容を変更する操作である。COMMITとABORTは、トランザクションの処理結果をコミットすなわち正しく完了した状態にするか、アボートすなわち処理結果をキャンセルして処理開始前の状態にする操作である。

【0052】図4のXMLドキュメントに対して、3つのトランザクション (txid=1のトランザクション (1)、txid=2のトランザクション (2)、txid=3のトランザクション (3) とする) が、以下に示すように並行にアクセスする場合の動作を例に、本実施形態のトランザクション処理システムの動作を説明する。

- [1] UPDATE (1, ICECREAM, 450円)
- [2] UPDATE (3, APPLE, 500円)
- [3] REMOVE (2, BISCUIT)
- [4] UPDATE (2, CHOCOLATE, 300円)



13

```

[5]  REMOVE (1, GRAPE)
[6]  UPDATE (2, KIWIFRUIT, 300円)
[7]  COMMIT (3)
[8]  COMMIT (1)
[9]  ABORT (2)

```

本トランザクション処理システムが図4のファイルを更新する場合、まず、共有データ管理部4から指示されたファイルシステム6が、ハードディスク10にあるファイルの内容をバッファ領域62に読み出す。トランザクションがコミットするまでは、ファイルの更新はバッファ領域62にあるコピーに対してのみ行い、ハードディスク10には書き戻さない。

【0053】[1] 最初に、UPDATE (1, ICE CREAM, 450円) で、トランザクション (1) がキー「ICE CREAM」を追加してその値を「450円」にする。このとき、ファイルがまだバッファ領域62に無ければ読み込む。その後、バッファ上のデータに対して、図5に示すようにバッファ領域62のデータを変更する。ここでは、新しい「ICE CREAM」をキーにするエントリを追加し、その値を「450円」にする。さらに、このエントリがトランザクション (1) によって更新中であることを示すために、<entry>タグの属性としてTXIDとTXSTATEの2つを記録しておく。属性TXIDの値には変更を行ったトランザクションのトランザクション識別子を、属性TXSTATEには変更の種類を記録しておき、COMMITあるいはABORTの処理時にこの情報を使用する。ここでは、TXID属性の値としてトランザクション識別子「1」と、TXSTATE属性の値として更新中を示す「UPDATE」を持たせる。

【0054】[2] 続いて、UPDATE (3, APPLE, 500円) で、トランザクション (3) がキー「APPLE」の値を「500円」に変更する。このときは、バッファ領域62のデータを図6に示すように変更する。ここでは、「APPLE」をキーにするエントリの<value>タグを<oldvalue>タグに変更して残しておき、新しい値「500円」を<value>タグで囲ったエレメントを作成してエントリに追加する。さらに、エントリのには、TXID属性の値としてトランザクション識別子「3」と、TXSTATE属性の値として更新中を示す「UPDATE」を持たせる。

【0055】[3] 次に、REMOVE (2, BISCUIT) で、トランザクション (2) がキー「BISCUIT」のエントリを削除する。このときは、バッファ領域62のデータを図7に示すように変更する。ここでは、「BISCUIT」をキーにするエントリの、TXID属性の値にトランザクション識別子「2」を、TXSTATE属性の値に削除されたことを示す「REMOVE」を持たせる。

14

【0056】[4] 次に、UPDATE (2, CHOCOLATE, 300円) でトランザクション (2) がキー「CHOCOLATE」の値を300円に変更した後のバッファ領域62のデータは、これまでと同様の処理を行って図8のようになる。

【0057】[5] 次に、REMOVE (1, GRAPE) でトランザクション (1) がキー「GRAPE」のエントリを削除した後のバッファ領域62のデータは、これまでと同様の処理を行って図9のようになる。

【0058】[6] 次に、UPDATE (2, KIWIFRUIT, 300円) でトランザクション (2) がキー「KIWIFRUIT」に対応するエントリを追加してその値を「300円」にした後のバッファ領域62のデータは、これまでと同様の処理を行って図10のようになる。

【0059】[7] 続いて、COMMIT (3) でトランザクション (3) のコミット処理が起動される。

【0060】このように複数のトランザクションが同じファイルを更新している場合のコミット処理の方式について説明する。

【0061】図10のようにファイルのデータがバッファ領域62上で変更されている状態で、COMMIT (3) でトランザクション (3) のコミット処理が起動されたとする。このとき、図11に示すように、トランザクション (3) による変更を、エレメントにTXIDやTXSTATEがついたり<oldvalue>タグを持っている一時的な状態から、それらを削除してハードディスク10に書き戻しても良い正しく更新された状態に変更する。こうしてトランザクション (3) の変更を反映したファイルのデータができると、それを書き戻してハードディスク10にある元のファイルを置き換える。このとき、本実施形態では、トランザクション (1) やトランザクション (2) による変更途中の状態のデータは、その変更途中のままハードディスク10に書き戻すことができる (後述するように、変更途中の状態のデータにはその更新を取り消すのに必要な情報を付加しているので、次に読み出したときに更新を取り消すための処理を行うことができる)。

【0062】[8] 次に、COMMIT (1) でトランザクション (1) のコミット処理が起動されたとする。このとき、図12に示すように、トランザクション (1) による変更を反映させ、それをハードディスク10に書き戻す。

【0063】[9] 次に、ABORT (2) でトランザクション (2) がアボートしたとする。このとき図13

15

に示すように、バッファ領域62にあるファイルのデータからトランザクション(2)による変更を取り消す。

【0064】以下では、これまでの例で説明した各々の操作の処理手順を説明する。

【0065】まず、READの処理手順について説明する。

【0066】READの操作は指定されたキーに対応するエントリの値を読み出すだけなので簡単である。ただし、READの実装には、そのシステムの使い方に応じて2通りの場合がある。他のトランザクションが更新中  
10 でなければ値を読み出せるという点に関しては共通であるが、あるトランザクションが読み出したエントリを、そのトランザクションが終了する前に他のトランザクションが更新できるか否かで2通りの方式に分かれる。つまり、1つの方式は、更新できるという解釈で、READに関しては何の排他制御もしないという方式である。もう1つの方式は、更新できないと言う解釈で、広く知られたREAD/WRITELockのセマンティクスに従って排他制御する方式である。すなわち、同じトランザクションによるREADと更新は両立するが、異なる  
20 トランザクションによるREADと更新は両立せず、また、異なるトランザクションのREADどうしは両立するというものである。排他制御をする後者の方式は、排他制御をしない前者の方式を包含するので、ここでは後者の方式の場合の処理手順を説明する。排他制御しない前者の方式は、後者の方式においてREADについては何も制御しないのと等価である。

【0067】図14にREAD(txid, key)の処理手順の一例を示す。

【0068】まず、キーの値がkeyに指定されたもの  
30 と同じエントリを探す(ステップS11)。

【0069】エントリが見つかったならば(ステップS12)、そのエントリにTXSTATE属性が付いているか否か、またTXSTATE属性が付いているならばそのTXSTATE属性が「READ」「UPDATE」「REMOVE」のいずれであるかを調べる(ステップS13)。

【0070】TXSTATE属性が付いていない場合は、誰もアクセスしていないので、そのエントリにTXID属性とTXSTATE属性を付け、TXID属性の  
40 値は「txid」に、TXSTATE属性の値は「READ」にし(ステップS14)、<value>部の値を結果として返す(ステップS15)。

【0071】TXSTATE属性の値が「READ」の場合は、そのTXID属性にまだ「txid」が入っていないければ追加し(ステップS16、S17)、<value>部の値を結果として返す(ステップS18)。

【0072】TXSTATE属性の値が「UPDATE」の場合は、同じトランザクションが変更した値であれば読めるので(ステップS19)、<value>部  
50

16

の値を結果として返す(ステップS20)。そうでない場合には、他のトランザクションが使用中で読めないことを知らせる。

【0073】TXSTATE属性の値が「REMOVE」ならば(ステップS13)、無いものを読もうとしているのでエラーになる。

【0074】なお、エントリが見つからなかったならば(ステップS12)、エラーとなる。

【0075】図15に、図4の状態のファイルのデータ  
10 に対して、READ(4, CHOCOLATE)でトランザクション4がキー「CHOCOLATE」の値を読み出した直後のバッファ領域62上のデータの状態を示す。

【0076】通常、TXIDの値はトランザクション識別子を1つ持っているが、TXSTATE属性の値が「READ」の場合のTXID属性にはトランザクション識別子のリストを持つことが出来る。これは、複数のトランザクションが同時に読み出すことが出来るためである。例えば、トランザクション識別子が1と4と5の  
20 トランザクションがREADしている場合、そのTXID属性の値は「1 4 5」のようなリストになっている。

【0077】次に、UPDATEの処理手順について説明する。

【0078】図16に、UPDATE(txid, key, value)の処理手順の一例を示す。

【0079】まず、キーの値がkeyに指定されたものと同じエントリを探す(ステップS21)。

【0080】エントリが見つかったならば(ステップS22)、そのエントリにTXSTATE属性が付いているか否か、またTXSTATE属性が付いているならばそのTXSTATE属性が「READ」「UPDATE」「REMOVE」のいずれであるかを調べる(ステップS23)。

【0081】TXSTATE属性が付いていない場合は、誰もアクセスしていないので、そのエントリにTXID属性とTXSTATE属性を付け、TXID属性の  
40 値は「txid」に、TXSTATE属性の値は「UPDATE」にし、現在の<value>部のタグ名を<oldvalue>に変更して退避し、新しく<value>部を作ってその値をUPDATEの第3引数のvalueに指定されたものにする(ステップS24)。

【0082】TXSTATE属性の値が「READ」の場合は、同じトランザクションだけがREADしている  
50 場合に限って更新できる。この場合は(ステップS25)、TXSTATE属性の値を「UPDATE」に変更し、現在の<value>部のタグ名を<oldvalue>に変更して退避し、新しく<value>部を作ってその値をUPDATEの第3引数のvalueに指定されたものにする(ステップS26)。他のトラン

17

ザクションがREAD中であれば、UPDATEすることができないので、使用中であることを知らせる。

【0083】TXSTATE属性の値が「UPDATE」の場合は、同じトランザクションが更新したエントリであれば（ステップS27）、その<value>部の値をUPDATEの第3引数のvalueに指定されたものに変更する（ステップS28）。他のトランザクションがUPDATE中であればUPDATEすることができないので、使用中であることを知らせる。

【0084】TXSTATE属性の値が「REMOVE」の場合は（ステップS23）、無いものを更新しようとしているのでエラーになる。

【0085】なお、キーの値がkeyに指定されたものと同じエントリが見つからない場合は（ステップS22）、新しくエントリを作成し、<key>部の値と<value>部の値を、それぞれUPDATEの第2引数と第3引数に指定されたものにしたエントリを作成し、そのTXID属性の値を「txid」に、TXSTATE属性の値を「UPDATE」にする（ステップS29）。

【0086】次に、REMOVEの処理手順について説明する。

【0087】図17に、REMOVE(txid, key)の処理手順の一例を示す。

【0088】まず、キーの値がkeyに指定されたものと同じエントリを探す（ステップS31）。

【0089】エントリが見つかったならば（ステップS32）、そのエントリにTXSTATE属性が付いているか否か、またTXSTATE属性が付いているならばそのTXSTATE属性が「READ」「UPDATE」「REMOVE」のいずれであるかを調べる（ステップS33）。

【0090】TXSTATE属性が付いていない場合は、誰もアクセスしていないので、TXID属性の値を「txid」に、TXSTATE属性の値を「REMOVE」にする（ステップS34）。

【0091】TXSTATE属性の値が「READ」の場合には、同じトランザクションだけがREADしている場合に限ってREMOVEできる。この場合は（ステップS35）、TXSTATE属性の値を「REMOVE」に変更する（ステップS36）。他のトランザクションがREAD中であればREMOVEすることができないので、使用中であることを知らせる。

【0092】TXSTATE属性の値が「UPDATE」の場合は、同じトランザクションが更新したエントリであれば（ステップS37）、REMOVEできる。このとき、<oldvalue>部があるかどうかで、そのエントリがこのトランザクションで新しく作られたものか、以前から存在していたものかがわかる。<oldvalue>部が無い場合には（ステップS38）、

18

このトランザクションが作成したエントリなので、即座にエントリを削除してしまっても良い（ステップS40）。そうでない場合には（ステップS38）、TXSTATE属性の値を「REMOVE」に変更する（ステップS39）。他のトランザクションがUPDATE中であれば（ステップS37）、REMOVEすることができないので、使用中であることを知らせる。

【0093】TXSTATE属性の値が「REMOVE」の場合は（ステップS33）、無いものを削除しようとしているのでエラーになる。

【0094】なお、エントリが見つからなかったならば（ステップS32）、エラーとなる。

【0095】次に、COMMITの処理手順について説明する。

【0096】図18に、COMMIT(txid)の処理手順の一例を示す。

【0097】COMMITの処理時には、ファイル中のすべてのエントリに対して図18の手順で処理を行う。

【0098】まず、そのエントリのTXSTATE属性の値を調べる（ステップS101）。

【0099】TXSTATE属性を持たない場合は、どのトランザクションもアクセスしていないので何もなくて良い。

【0100】TXSTATE属性が「READ」の場合には、そのTXID属性の中にtxidが含まれていて（ステップS102）、txid以外のトランザクション識別子も含まれていれば（ステップS103）、txidを抜く（ステップS104）。TXID属性がtxidのみをもっている場合には（ステップS103）、TXID属性とTXSTATE属性を削除する（ステップS105）。

【0101】次に、TXSTATE属性の値が「UPDATE」の場合は、そのTXID属性の値がtxidと同じであれば（ステップS106）、<oldvalue>部とTXID属性とTXSTATE属性を削除してその更新を有効にする（ステップS107）。

【0102】TXSTATE属性がREMOVEの場合には（ステップS101）、そのTXID属性の値がtxidと同じであれば（ステップS108）、そのエントリを削除する（ステップS109）。

【0103】すべてのエントリに関して以上の処理が終了すれば、その時点のバッファ領域62上のデータをハードディスク10上に書き戻して、ファイルを更新する。

【0104】ところで、COMMIT処理の最後に、共有データ管理部4は、バッファ領域62上で更新したファイルをハードディスク10に書き込むように、ファイルシステム6に指示する。このとき、このファイルの書き込みは、コミットするトランザクションが更新した他のファイルと同期してアトミックに書き込む必要があ

る。すなわち、トランザクションを実行するアプリケーションプログラム2は、共有データ管理部4を介さずにファイルシステム6に直接指示して複数のファイルの更新を行っている可能性があり、また、共有データ管理部4を介しても複数のファイルを更新している可能性がある(図2および図3参照)。トランザクションとしてアトミックにコミットするためには、これらの複数のファイルへの更新がすべて完全に行われるなければならない。本実施形態のトランザクション処理システムでは、ファイルシステム6は、トランザクション管理表61によって、アプリケーションプログラム2から直接指示されて更新したファイルのリストをトランザクション毎に管理しており、その情報を用いて、同じトランザクションが更新したファイルを一括してアトミックにハードディスク10に書き戻すことができる。

【0105】しかし、共有データ管理部4を介して操作しているファイルに関しては、ファイルシステム6内ではどのトランザクションとも関連しないファイルとして管理されている。それらのファイルをどのトランザクションのコミット時に書き戻せば良いかは、共有データ管理部4がCOMMITの処理時に判断する。

【0106】そこで、共有データ管理部4がどのトランザクションとも関連せずにオープンして更新したファイルに対して、特定のトランザクションのコミット時に同期してアトミックにハードディスク10に書き戻すことを指示する機能を、ファイルシステム6に持たせる。COMMITの処理を行う共有データ管理部4は、指定されたトランザクション識別子のトランザクションによる更新をファイルに反映させた後、そのファイルを指定されたトランザクション識別子のトランザクションと同期してハードディスク10に書き戻すように、ファイルシステム6に指示する。ファイルシステム6は、このようにして指定されたファイルを、指定されたトランザクション識別子のトランザクションのコミット時に、トランザクション管理表61にあるそのトランザクションが更新したファイルと同時に、アトミックにハードディスク10に書き戻す。

【0107】ただし、ファイルシステム6のコミット処理が終了後は、トランザクションを実行するアプリケーションプログラム2が直接ファイルシステム6に指示して更新したファイルの情報(トランザクション管理表61のエントリやバッファ領域62にあるファイルのイメージ)は不要になる。一方、共有データ管理部4が特定のトランザクションに関連せずに更新しているファイルは、その後も他のトランザクションによって更新が続けられるため、どのトランザクションとも関連しない状態のまま共有データ管理部4によって使い続けられる。そのため、ファイルシステム6はそれらのファイルに関する情報は維持しつづける。

【0108】次に、ABORTの処理手順について説明

する。

【0109】図19に、ABORT(txid)の処理手順の一例を示す。

【0110】ABORTの処理時には、すべてのエントリに対して図19の手順で処理を行う。

【0111】まず、そのエントリのTXSTATE属性の値を調べる(ステップS111)。

【0112】TXSTATE属性を持たない場合は、どのトランザクションもアクセスしていないので何もしなくて良い。

【0113】TXSTATE属性が「READ」の場合は、そのTXID属性の中にtxidが含まれていて(ステップS112)、txid以外のトランザクション識別子も含まれていれば(ステップS113)、txidを抜く(ステップS104)。TXID属性がtxidのみをもっている場合には(ステップS113)、TXID属性とTXSTATE属性を削除する(ステップS115)。

【0114】TXSTATE属性の値が「UPDATE」の場合は、そのTXID属性の値がtxidと同じであれば(ステップS116)、そのエントリのUPDATEを取り消す操作を行う。このとき、そのエントリに<oldvalue>部があれば(ステップS117)、現在の<value>部を削除して<oldvalue>部のタグ名を<value>に戻すことで更新前の値に戻すとともに、TXID属性とTXSTATE属性を削除する(ステップS118)。そのエントリに<oldvalue>部が無い場合は(ステップS117)、このトランザクションで新しく作成したエントリであるので、エントリを削除する(ステップS119)。

【0115】TXSTATE属性がREMOVEの場合は(ステップS111)、そのTXID属性の値がtxidと同じであれば(ステップS120)、TXID属性とTXSTATE属性を削除して、REMOVEを取り消す(ステップS121)。

【0116】ところで、ABORTの場合、アボートするトランザクションによるファイルの更新を、バッファ領域62上で取り消す。このとき、同じファイルを更新していて既にコミットした他のトランザクションがあった場合には、アボートするトランザクションによる更新中の状態もファイルに書きこまれてしまっている可能性がある。しかし、ABORT処理時にバッファ領域62上で取り消した結果のファイルをハードディスク10に書き込むことは、必ずしも必要無い。なぜなら、次に同じファイルを他のトランザクションが更新してコミットすれば、アボートしたトランザクションによる更新の取り消しはハードディスク10上に正しく反映される。また、たとえ次にどのトランザクションも更新せずに、そのまま計算機の障害等で忘れ去られてしまった場合で

21

も、後で述べるリカバリ処理によって、次にそのファイルをオープンする時に、ハードディスク 10 上のファイルに残っているコミットしていないトランザクションによる更新途中の状態は取り消される。

【0117】次に、リカバリの処理手順について説明する。

【0118】本実施形態のトランザクション処理システムでは、コミットしていないトランザクションによって更新途中のファイルの状態がハードディスク 10 に書き戻される。そのため、障害で計算機がダウンした場合には、更新途中の状態が残るので、リカバリ処理を行って更新途中で残された変更を取り消さなくてはならない。従来のトランザクション処理方式では、このようなリカバリ処理は障害が起こって再起動した直後にまとめて行わなければならない。そのため、障害直後の再起動のオーバーヘッドは大きい。しかし、本実施形態のトランザクション処理方式では、リカバリ処理は、次にそのファイルをハードディスク 10 から読み出す時点で良い。

【0119】図 20 に、リカバリの処理手順の一例を示す。

【0120】共有データ管理部 4 がまだバッファ領域 62 にないファイルを参照するためにオープン処理をファイルシステム 6 に指示すると、ファイルシステム 6 はそのファイルをハードディスク 10 から読み出してバッファ領域 62 にコピーする。その時点で共有データ管理部 4 は、すべてのエントリに対して図 20 に示す手順でリカバリ処理を実行する。

【0121】まず、そのエントリの TXSTATE 属性を調べる (ステップ S201)。

【0122】TXSTATE 属性を持たない場合は、どのトランザクションもアクセスしていないので何もしなくて良い。

【0123】TXSTATE 属性の値が「READ」の場合は、TXID 属性と TXSTATE 属性を削除するだけで良い (ステップ S202)。

【0124】TXSTATE 属性の値が「UPDATE」の場合は、いずれかのトランザクションが更新している途中で障害が発生したことを示しているので、それらをアボートする。このときの動作は、そのエントリに <oldvalue> 部があるかどうかで異なる。<oldvalue> 部があれば (ステップ S203)、そのトランザクションの開始以前から存在していたエントリであるので、まず現在の <value> 部を削除し、<oldvalue> 部のタグ名を <value> に変更して値を元に戻し、さらに TXID 属性と TXSTATE 属性を削除する (ステップ S204)。<oldvalue> 部が無い場合は (ステップ S203)、このトランザクションが作成したエントリであるので、エントリ全体を削除する (ステップ S205)。

【0125】TXSTATE 属性の値が「REMOV

22

E」の場合は (ステップ S201)、TXID 属性と TXSTATE 属性を削除するだけで REMOVE をキャンセルすることが出来る (ステップ S206)。

【0126】ところで、ファイルをディスクから読み出す度にこのようなリカバリ処理を行うのでは、非常にオーバーヘッドが大きい。そこで、ファイルを書き出すときにそのファイルが更新途中の状態を含んでいるかどうかを簡単に識別できる情報を付けておくことで、このオーバーヘッドを無くすことが出来る。その方式は、例えば図 21 に示すように、ドキュメントのルートのエレメントの属性として、現在いくつかのトランザクションがファイルを更新中かを示す情報を持たせる。図 21 の例では、ルートである <KVtable> の TXNUM という属性がそれを示しており、現在、トランザクション (1) とトランザクション (2) の 2 つのトランザクションが更新中であることを示している。こうしておく

と、最初にファイルをディスクから読みだして来た時に、そのルートエレメントが TXNUM という属性を持っていたら、そのファイル内のすべてのエントリに対して図 20 のリカバリ処理を実行する。もし TXNUM という属性が無ければ、リカバリ処理は必要無い。

【0127】さて、これまでの説明では、エントリはルートエレメントである <KVtable> の下にフラットに記録していたが、例えば良く知られた二分木のような構造を使って、検索を高速化するように実施することができる。図 22 は、そのように実施した場合のファイルのデータの例である。この例では、エントリには <key> と <value> 以外に、<less> と <greater> というエレメントを持つ。そのエレメントのキーより小さなキーを持つエントリは <less> の下に、大きなキーを持つエントリは <greater> の下にくるように、二分木を構成している。この場合も、トランザクションによる UPDATE や REMOVE などの更新はフラットな場合と同様に扱うことが出来る。ただし、エントリの追加や削除によって木の構成を組み換える処理が必要になる。この処理は通常の木構造のノードの追加/削除とまったく同じ処理である。木構造の組み換え時には、AVL 木のような手法を使ってうまく木がバランスするように組み換えたり、アクセス頻度の多いエントリが木のルートに近い場所に来るように組み換えたりするように実施することもできる。

【0128】また、本発明は、良く知られた B-TREE のような管理構造を使って、ファイルを分割して管理するように実施することも出来る。図 23/図 24/図 25/図 26 にその例を示す。ここでは、一つのファイルには最大 6 個のエントリを収容し、それ以上にエントリが増えるとファイルを分割するというような、B-TREE と同じアルゴリズムでエントリの追加/削除の処理を行うものとする。図 23/図 24/図 25/図 26 の例では、FILE001 がルートになるファイル、F

23

FILE002, FILE003, FILE004がその下にあるファイルである。

【0129】図27に、このように一纏まりの情報を複数のファイルに分割して記録・管理するようにしたトランザクション処理システムの構成例を示す。

【0130】図27の構成は、図1の構成と比べて、共有データ管理部4が更新グループ表42を持つ点が異なる。後に説明するように、複数のファイル間で情報を移動してファイル間の構造を変更した場合に、それらのファイルをコミット時に同時にハードディスク10に書き戻さなければならない。更新グループ表42は、そのようなコミット時に同時にハードディスク10に書き戻したりあるいは削除したりする必要のあるファイルのグループを管理するための表である。

【0131】図28に更新グループ表42の例を示す。この更新グループ表には、コミット時に同時に書き戻したり削除したりすべきファイルのグループ毎に、そのメンバが記録されている。図28では、例えば、FILE001とFILE002とFILE005がグループになっている。

【0132】さて、図23／図24／図25／図26のFILE001から始めて「SESAME」をキーとするエントリを探す手順は、FILE001のルートから始める。まずキーを「KIWIFRUIT」と比べるとそれよりも大きい。次にキーを<greater>側の「ROSEMARY」と比べるとそれよりも大きい。次に<greater>側を見るとFILE004へのリンクになっている。そこでFILE004に移って、キーを「THYME」と比べるとそれよりも小さい。そこで<less>側を見ると、キーが「SESAME」のエントリが見つかる。こうしてエントリが見つければ、あとはそれへの操作はこれまでの実施形態と同様に、TXID属性とTXSTATE属性をつけて更新の途中状態を回復可能な形式で保存して行く。

【0133】例えば、図23／図24／図25／図26の状態のデータに対して、トランザクション(1)がキー「ORANGE」のエントリの値を「250円」に更新し、さらにキー「SESAME」のエントリの値を「150円」に更新し、それと並行に動いているトランザクション(2)がキー「VODKA」のエントリを削除したとする。この状態で図23／図24／図25／図26のデータは図29／図30のような状態になる。UPDATEやREMOVEなどの更新途中の状態の持ち方は、一つのファイルにすべてのエントリを入れる場合と同じであり、更新したエントリにはTXIDとTXSTATEの2つの属性をつけて管理する。

【0134】図29／図30の状態、トランザクション(1)と(2)が順にコミットおよびアボートする場合の動作例を説明する。

【0135】まず、トランザクション(1)のコミット

24

時には、トランザクション(1)によって更新されたエントリはファイルFILE003とFILE004にのみ存在している。そこで、FILE003とFILE004に対してコミット処理を行った図31／図32の状態のデータを作成し、これをトランザクション(1)による他のファイルの更新と同時にハードディスク10に書き戻す。このとき、FILE004にはトランザクション(2)による更新途中の状態が含まれたまま書き戻すのは、これまでの実施形態と同じである。

【0136】次に、トランザクション(2)のアボート時には、トランザクション(2)によって更新されたエントリはFILE004にのみ存在している。そこでFILE004に対してアボート処理を行った図33の状態を作成して、アボート処理は終了する。アボート処理時にはファイルのハードディスク10への書き戻しは必要無いのは、これまでの実施形態と同じである。

【0137】さて、この状態で、新しくトランザクション(3)がキー「DODO」とキー「HOPOE」のエントリを追加し、それぞれの値を「5000000円」と「9800000円」にした後のデータの状態を、図34／図35／図36に示す。ここでは、一つのファイルに入れることの出来るエントリの数は決まっているので、この更新例のように一つのファイルにその容量以上のエントリを追加しようとした場合には、ファイル間で組み換えが発生してエントリが移動する。図34／図35／図36の例では、キー「DODO」とキー「HOPOE」の二つのエントリを追加しようとしたところ、それらはどちらもFILE002への追加となる。その結果FILE002の分割が行われて、新たにFILE005が作られてエントリが移動している。

【0138】このようにファイル間でエントリ情報の移動が発生したり、新しいファイルが追加されたり、あるいは逆にファイルが削除されたりすると、それらのファイルの更新は同時にハードディスク10へ反映させなければならない。そのために、本実施形態のトランザクション処理システムの共有データ管理部4は、更新グループ表42を管理している。図34／図35／図36の例のように、FILE002へのエントリの追加によってFILE002の分割が発生し、FILE001の更新とFILE005の作成が発生した場合、共有データ管理部4は更新グループ表42にFILE001とFILE002とFILE005をひとつのグループとして登録する。

【0139】トランザクション(3)のコミットを行う場合、トランザクション(3)が更新したエントリはFILE001とFILE005のみに存在するが、更新グループ表42を参照することで、FILE002も同時に書き戻さなければならないことがわかる。そこで、図37／図38に示すようにコミット処理を行ったFILE001とFILE005に加えてFILE002の

25

3つのファイルをハードディスク10に書き戻す。

【0140】このように複数のファイルに分割してデータを持つように本発明を実施する場合には、様々な基準で一つのファイルに入れられるエントリの数を制限することが出来る。そのような基準としては、図34/図35/図36の例のように一つのファイル内のエントリで制限する方式、ファイルの物理的なサイズで制限する方式、追加削除の頻度等の情報によって最大のエントリ数を変更する方式、それらの組み合わせなど、様々な方式が考えられる。

【0141】本実施形態のように複数のファイルに分割してデータを記憶する方式は、エントリの数が大きくなったときでも、その一部分しか更新していないなら、コミット時にディスクに書き出すデータ量を減らせるという効果がある。すべてのエントリを一つのファイルに入れる方式では、大きなファイルの一部しか変更していない場合でも、コミット時にファイル全体をディスクに書き戻す必要がある。

【0142】さて、これまでは、キーと値とのペアというデータベース的な構造を持つデータに対して本発明を適用する場合を例として説明してきた。以下では、他の例として、より一般的なXMLドキュメントに対して本発明を適用する場合について説明する。

【0143】図39は、一般的なXMLのドキュメントの例である。このドキュメントに対して、トランザクション(1)が「<title>名古屋</title>」の<chapter>の<author>を「味噌かつ」に変更し、トランザクション(2)が「<chapter><title>博多</title><author>からし明太子</author></chapter>」を追加した状態のデータが図40である。

【0144】この後、トランザクション(1)のコミット時にデータを図41の状態に更新し、これをディスクに書き戻す。

【0145】さらに、トランザクション(2)のコミット時にはデータを図42の状態に更新し、これをディスクに書き戻す。

【0146】この一般的なXMLドキュメントを対象にした実施形態でも、障害発生によって更新途中で残されたファイルは、次にディスクから読み出した時にチェックし、TXSTATEがADDのエレメントは削除してTXSTATEがREMOVEのエレメントは残すようになりカバリ処理を行うことで、コミットしていないトランザクションによる一時的な更新を取り消すことが出来る。

【0147】本実施形態によれば、並行に動作する複数のトランザクションによるファイルの更新を、簡単な処理手順で実現することができる。本実施形態によれば、従来のWAL方式や、シャドウページ方式の上に特殊なログ管理を実装するときのように、データの更新時やコ

26

ミット処理時あるいはリカバリ時に複雑な処理を必要としない。

【0148】また、本実施形態では、障害発生後のリカバリ処理を、再起動時に集中して行う必要は無く、障害で更新途中の状態は次にそのファイルを使う時点で行われるため、リカバリ処理のオーバーヘッドが小さい。

【0149】さらに、本実施形態では、1つのデータを複数のファイルに分けて管理することで、ファイルの更新のコミット時にディスクに書き戻すデータの量を減らすことができる。

【0150】なお、以上の各機能は、ソフトウェアとしても実現可能である。

【0151】また、本実施形態は、コンピュータに所定の手段を実行させるための(あるいはコンピュータに所定の手段として機能させるための、あるいはコンピュータに所定の機能を実現させるための)プログラムを記録したコンピュータ読取り可能な記録媒体としても実施することもできる。

【0152】本発明は、上述した実施の形態に限定されるものではなく、その技術的範囲において種々変形して実施することができる。

【0153】

【発明の効果】本発明によれば、並行に動作する複数のトランザクションによるファイルの更新処理と障害時のリカバリ処理を簡単に効率良く実現することができる。

【図面の簡単な説明】

【図1】本発明の一実施形態に係るトランザクション処理システムの構成例を示す図

【図2】更新管理表の一例を示す図

【図3】トランザクション管理表の一例を示す図

【図4】XMLドキュメント(データファイル)の一例を示す図

【図5】UPDATE(1, ICECREAM, 450円)の処理直後のデータの一例を示す図

【図6】UPDATE(3, APPLE, 500円)の処理直後のデータの一例を示す図

【図7】REMOVE(2, BISCUIT)の処理直後のデータの一例を示す図

【図8】UPDATE(2, CHOCOLATE, 300円)の処理直後のデータの一例を示す図

【図9】REMOVE(1, GRAPE)の処理直後のデータの一例を示す図

【図10】UPDATE(2, KIWIFRUIT, 300円)の処理直後のデータの一例を示す図

【図11】COMMIT(3)の処理直後のデータの一例を示す図

【図12】COMMIT(1)の処理直後のデータの一例を示す図

【図13】ABORT(2)の処理直後のデータの一例を示す図

【図14】READ (txid, key) の処理手順の一例を示すフローチャート

【図15】READ (4, CHOCOLATE) の処理直後のデータの一例を示す図

【図16】UPDATE (txid, key, value) の処理手順の一例を示すフローチャート

【図17】REMOVE (txid, key) の処理手順の一例を示すフローチャート

【図18】COMMIT (txid) の処理手順の一例を示すフローチャート

【図19】ABORT (txid) の処理手順の一例を示すフローチャート

【図20】リカバリの処理手順の一例を示すフローチャート

【図21】トランザクションによる更新中情報を持つデータの一例を示す図

【図22】二分木構造により検索を高速化するデータの一例を示す図

【図23】複数ファイルに分割したデータ管理方式の一例を示す図

【図24】複数ファイルに分割したデータ管理方式の一例を示す図

【図25】複数ファイルに分割したデータ管理方式の一例を示す図

【図26】複数ファイルに分割したデータ管理方式の一例を示す図

【図27】本発明の一実施形態に係るトランザクション処理システムの他の構成例を示す図

【図28】更新グループ表の一例を示す図

【図29】トランザクション (1) と (2) による更新の例を示す図

【図30】トランザクション (1) と (2) による更新の例を示す図

【図31】トランザクション (1) のコミット処理の例 \*

【図2】

ファイル	更新トランザクション
F0102	36,21,5
F2156	6,
F3624	23,6
⋮	
F7531	5,1

\*を示す図

【図32】トランザクション (1) のコミット処理の例を示す図

【図33】トランザクション (2) のアボート処理の例を示す図

【図34】トランザクション (3) による更新の例を示す図

【図35】トランザクション (3) による更新の例を示す図

10 【図36】トランザクション (3) による更新の例を示す図

【図37】トランザクション (3) のコミット処理の例を示す図

【図38】トランザクション (3) のコミット処理の例を示す図

【図39】一般的なXMLドキュメントの例を示す図

【図40】トランザクション (1) と (2) によるデータの更新例を示す図

20 【図41】トランザクション (1) コミット後のデータの一例を示す図

【図42】トランザクション (2) コミット後のデータの一例を示す図

【図43】従来のトランザクション処理システムの例を示す図

【符号の説明】

2…アプリケーションプログラム

4…共有データ管理部

41…更新管理表

42…更新グループ表

6…ファイルシステム

61…トランザクション管理表

62…バッファ領域

8…トランザクションマネージャ

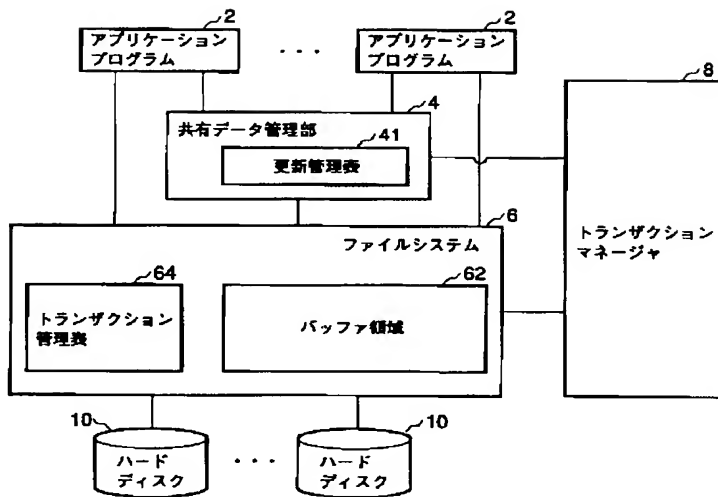
10…ハードディスク

【図3】

トランザクション	ファイル
3	F9268
5	F0100,F9264
6	F2164,F1234
21	F0104
⋮	
NOTX	F0102,F2156,F3624,F5497,F7531



【図1】



【図4】

```

<KVtable>
  <entry>
    <key>APPLE</key>
    <value>400円</value>
  </entry>
  <entry>
    <key>BISCUIT</key>
    <value>250円</value>
  </entry>
  <entry>
    <key>CHOCOLATE</key>
    <value>200円</value>
  </entry>
  <entry>
    <key>GRAPE</key>
    <value>600円</value>
  </entry>
</KVtable>

```

【図13】

【図5】

```

<KVtable>
  <entry>
    <key>APPLE</key>
    <value>400円</value>
  </entry>
  <entry>
    <key>BISCUIT</key>
    <value>250円</value>
  </entry>
  <entry>
    <key>CHOCOLATE</key>
    <value>200円</value>
  </entry>
  <entry>
    <key>GRAPE</key>
    <value>600円</value>
  </entry>
  <entry TXID="1" TXSTATE="UPDATE">
    <key>ICECREAM</key>
    <value>450円</value>
  </entry>
</KVtable>

```

```

<KVtable>
  <entry>
    <key>APPLE</key>
    <value>500円</value>
  </entry>
  <entry>
    <key>BISCUIT</key>
    <value>250円</value>
  </entry>
  <entry>
    <key>CHOCOLATE</key>
    <value>200円</value>
  </entry>
  <entry>
    <key>ICECREAM</key>
    <value>450円</value>
  </entry>
</KVtable>

```

【図6】

```

<KVtable>
  <entry TXID="3" TXSTATE="UPDATE">
    <key>APPLE</key>
    <value>500円</value>
    <oldvalue>400円</oldvalue>
  </entry>
  <entry>
    <key>BISCUIT</key>
    <value>250円</value>
  </entry>
  <entry>
    <key>CHOCOLATE</key>
    <value>200円</value>
  </entry>
  <entry>
    <key>GRAPE</key>
    <value>600円</value>
  </entry>
  <entry TXID="1" TXSTATE="UPDATE">
    <key>ICECREAM</key>
    <value>450円</value>
  </entry>
</KVtable>

```

【図7】

```

<KVtable>
  <entry TXID="3" TXSTATE="UPDATE">
    <key>APPLE</key>
    <value>500円</value>
    <oldvalue>400円</oldvalue>
  </entry>
  <entry TXID="2" TXSTATE="REMOVE">
    <key>BISCUIT</key>
    <value>250円</value>
  </entry>
  <entry>
    <key>CHOCOLATE</key>
    <value>200円</value>
  </entry>
  <entry>
    <key>GRAPE</key>
    <value>600円</value>
  </entry>
  <entry TXID="1" TXSTATE="UPDATE">
    <key>ICECREAM</key>
    <value>450円</value>
  </entry>
</KVtable>

```

【図8】

```

<KVtable>
  <entry TXID="3" TXSTATE="UPDATE">
    <key>APPLE</key>
    <value>500円</value>
    <oldvalue>400円</oldvalue>
  </entry>
  <entry TXID="2" TXSTATE="REMOVE">
    <key>BISCUIT</key>
    <value>250円</value>
  </entry>
  <entry TXID="2" TXSTATE="UPDATE">
    <key>CHOCOLATE</key>
    <value>300円</value>
    <oldvalue>200円</oldvalue>
  </entry>
  <entry>
    <key>GRAPE</key>
    <value>600円</value>
  </entry>
  <entry TXID="1" TXSTATE="UPDATE">
    <key>ICECREAM</key>
    <value>450円</value>
  </entry>
</KVtable>

```

【図9】

```

<KVtable>
  <entry TXID="3" TXSTATE="UPDATE">
    <key>APPLE</key>
    <value>500円</value>
    <oldvalue>400円</oldvalue>
  </entry>
  <entry TXID="2" TXSTATE="REMOVE">
    <key>BISCUIT</key>
    <value>250円</value>
  </entry>
  <entry TXID="2" TXSTATE="UPDATE">
    <key>CHOCOLATE</key>
    <value>300円</value>
    <oldvalue>200円</oldvalue>
  </entry>
  <entry TXID="1" TXSTATE="REMOVE">
    <key>GRAPE</key>
    <value>600円</value>
  </entry>
  <entry TXID="1" TXSTATE="UPDATE">
    <key>ICECREAM</key>
    <value>450円</value>
  </entry>
</KVtable>

```

【図10】

```

<KTable>
<entry TXID="3" TXSTATE="UPDATE">
  <key>APPLE</key>
  <value>500円</value>
  <oldvalue>400円</oldvalue>
</entry>
<entry TXID="2" TXSTATE="REMOVE">
  <key>BISCUIT</key>
  <value>250円</value>
</entry>
<entry TXID="2" TXSTATE="UPDATE">
  <key>CHOCOLATE</key>
  <value>300円</value>
  <oldvalue>200円</oldvalue>
</entry>
<entry TXID="1" TXSTATE="REMOVE">
  <key>GRAPE</key>
  <value>600円</value>
</entry>
<entry TXID="1" TXSTATE="UPDATE">
  <key>ICECREAM</key>
  <value>450円</value>
</entry>
<entry TXID="2" TXSTATE="UPDATE">
  <key>KIWIFRUIT</key>
  <value>300円</value>
</entry>
</KTable>

```

【図11】

```

<KTable>
<entry>
  <key>APPLE</key>
  <value>500円</value>
</entry>
<entry TXID="2" TXSTATE="REMOVE">
  <key>BISCUIT</key>
  <value>250円</value>
</entry>
<entry TXID="2" TXSTATE="UPDATE">
  <key>CHOCOLATE</key>
  <value>300円</value>
  <oldvalue>200円</oldvalue>
</entry>
<entry TXID="1" TXSTATE="REMOVE">
  <key>GRAPE</key>
  <value>600円</value>
</entry>
<entry TXID="1" TXSTATE="UPDATE">
  <key>ICECREAM</key>
  <value>450円</value>
</entry>
<entry TXID="2" TXSTATE="UPDATE">
  <key>KIWIFRUIT</key>
  <value>300円</value>
</entry>
</KTable>

```

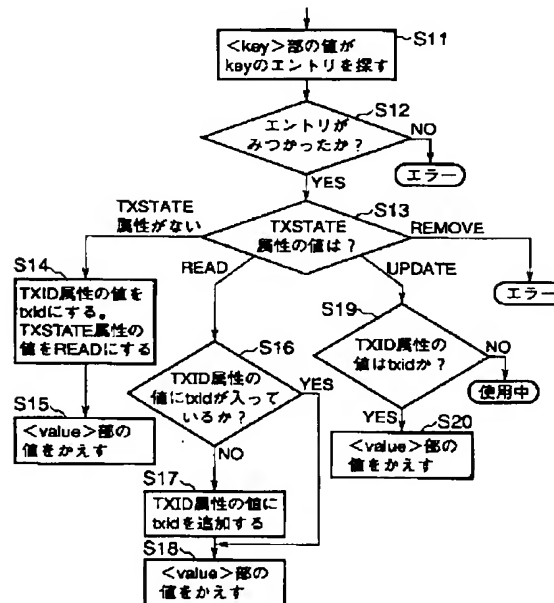
【図12】

```

<KTable>
<entry>
  <key>APPLE</key>
  <value>500円</value>
</entry>
<entry TXID="2" TXSTATE="REMOVE">
  <key>BISCUIT</key>
  <value>250円</value>
</entry>
<entry TXID="2" TXSTATE="UPDATE">
  <key>CHOCOLATE</key>
  <value>300円</value>
  <oldvalue>200円</oldvalue>
</entry>
<entry>
  <key>ICECREAM</key>
  <value>450円</value>
</entry>
<entry TXID="2" TXSTATE="UPDATE">
  <key>KIWIFRUIT</key>
  <value>300円</value>
</entry>
</KTable>

```

【図14】



【図28】

メンバファイル
FILE001,FILE002,FILE005
FILE125,FILE126,FILE131,FILE132
FILE256,FILE328,FILE549
⋮

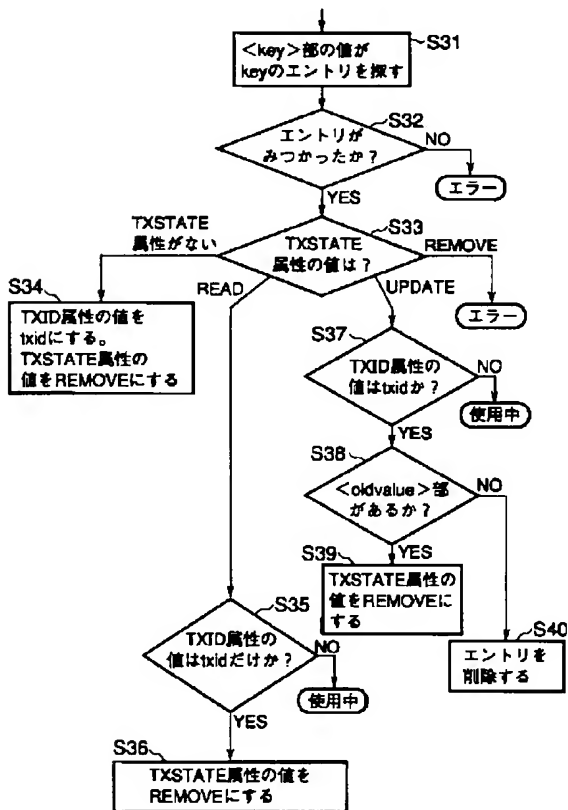
【図15】

```

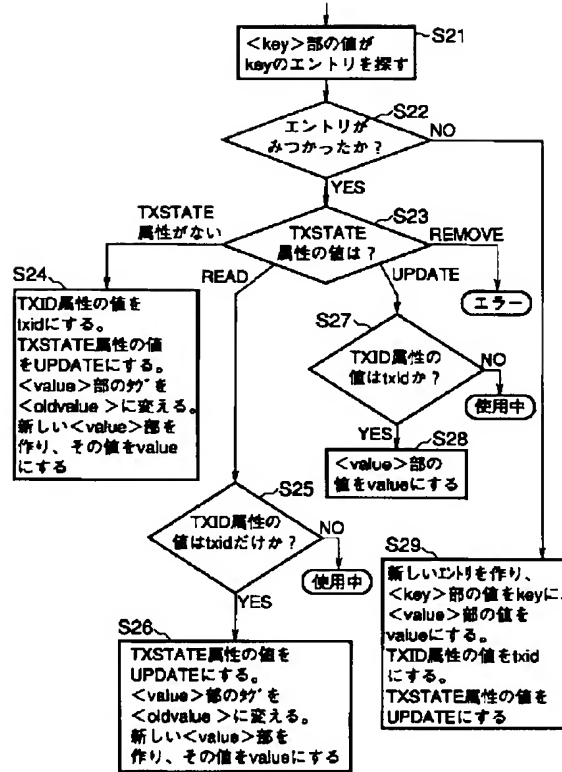
<KTable>
<entry>
<key>APPLE</key>
<value>400円</value>
</entry>
<entry>
<key>BISCUIT</key>
<value>250円</value>
</entry>
<entry TXID="4" TXSTATE="READ">
<key>CHOCOLATE</key>
<value>200円</value>
</entry>
<entry>
<key>GRAPE</key>
<value>600円</value>
</entry>
</KTable>

```

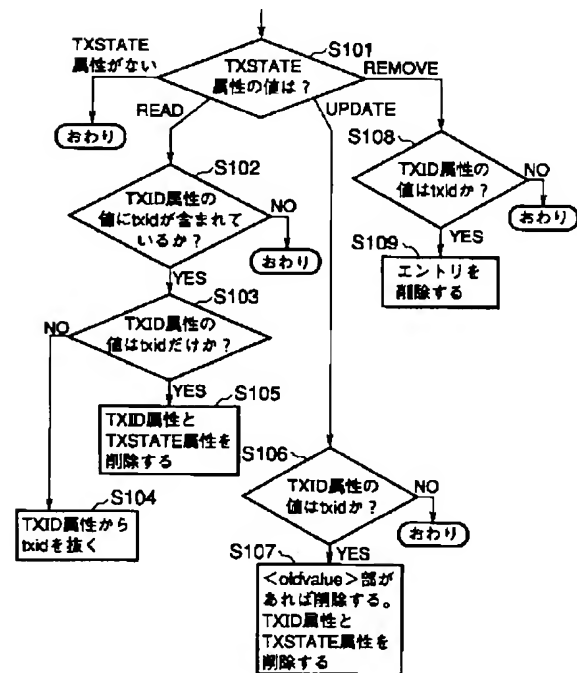
【図17】



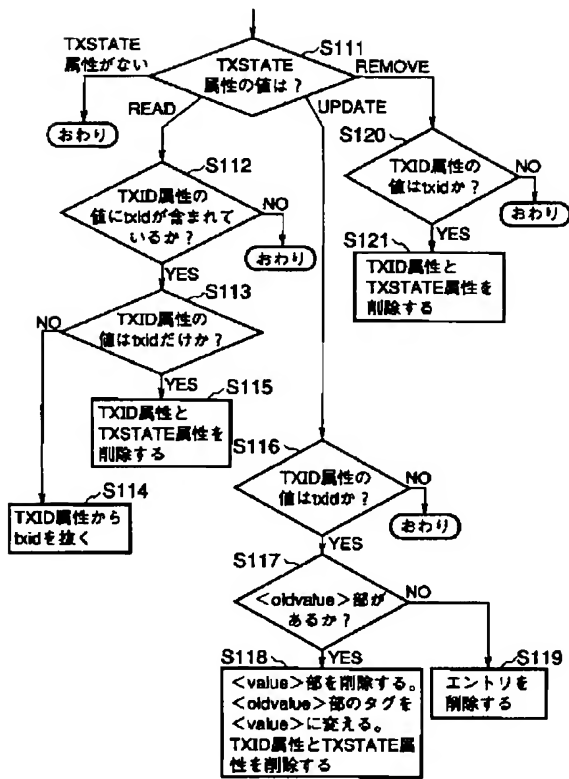
【図16】



【図18】



【図19】

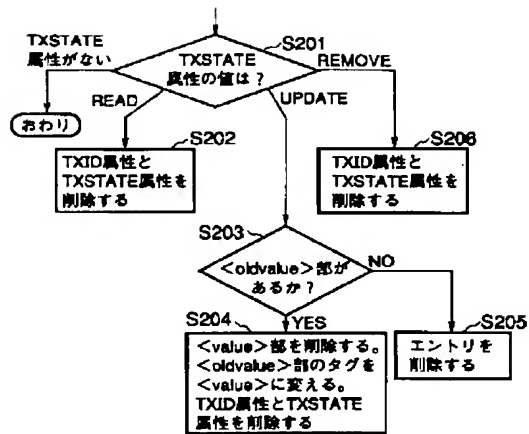


【図21】

```

<KVtable TXNUM="2">
  <entry>
    <key>APPLE</key>
    <value>500円</value>
  </entry>
  <entry TXID="2" TXSTATE="REMOVE">
    <key>BISCUIT</key>
    <value>250円</value>
  </entry>
  <entry TXID="2" TXSTATE="UPDATE">
    <key>CHOCOLATE</key>
    <value>300円</value>
    <oldvalue>200円</oldvalue>
  </entry>
  <entry TXID="1" TXSTATE="REMOVE">
    <key>GRAPE</key>
    <value>600円</value>
  </entry>
  <entry TXID="1" TXSTATE="UPDATE">
    <key>ICECREAM</key>
    <value>450円</value>
  </entry>
  <entry TXID="2" TXSTATE="UPDATE">
    <key>KIWIFRUIT</key>
    <value>300円</value>
  </entry>
</KVtable>
  
```

【図20】



【図22】

```

<KVlist>
  <entry>
    <key>GRAPE</key>
    <value>600円</value>
  </entry>
  <less>
    <entry>
      <key>BISCUIT</key>
      <value>250円</value>
    </entry>
    <entry>
      <key>APPLE</key>
      <value>400円</value>
    </entry>
  </less>
  <greater>
    <entry>
      <key>CHOCOLATE</key>
      <value>200円</value>
    </entry>
    <greater>
      <entry>
        <key>ICECREAM</key>
        <value>450円</value>
      </entry>
    </greater>
  </greater>
  <greater>
    <entry>
      <key>KIWIFRUIT</key>
      <value>300円</value>
    </entry>
    <less>
      <entry>
        <key>LEMON</key>
        <value>100円</value>
      </entry>
    </less>
  </greater>
</KVlist>
  
```

【図 23】

```

[FILE001]
<KVlist>
  <entry>
    <key>KIWIFRUIT</key>
    <value>300円</value>
    <less>
      <link>FILE002</link>
    </less>
    <greater>
      <entry>
        <key>ROSEMARY</key>
        <value>180円</value>
      </entry>
    </greater>
    <less>
      <link>FILE003</link>
    </less>
    <greater>
      <link>FILE004</link>
    </greater>
  </entry>
</KVlist>

```

【図 25】

```

[FILE003]
<KVlist>
  <entry>
    <key>MINT</key>
    <value>120円</value>
    <less>
      <entry>
        <key>LEMON</key>
        <value>100円</value>
      </entry>
    </less>
    <greater>
      <entry>
        <key>ORANGE</key>
        <value>210円</value>
      </entry>
    </greater>
    <entry>
      <key>PINEAPPLE</key>
      <value>980円</value>
    </entry>
  </greater>
</entry>
</KVlist>

```

【図 24】

```

[FILE002]
<KVlist>
  <entry>
    <key>CHOCOLATE</key>
    <value>200円</value>
    <less>
      <entry>
        <key>APPLE</key>
        <value>400円</value>
      </entry>
    </less>
    <greater>
      <entry>
        <key>BISCUIT</key>
        <value>250円</value>
      </entry>
    </greater>
  </entry>
  <less>
    <greater>
      <entry>
        <key>GRAPE</key>
        <value>600円</value>
      </entry>
    </greater>
    <entry>
      <key>ICECREAM</key>
      <value>450円</value>
    </entry>
  </greater>
</entry>
</KVlist>

```

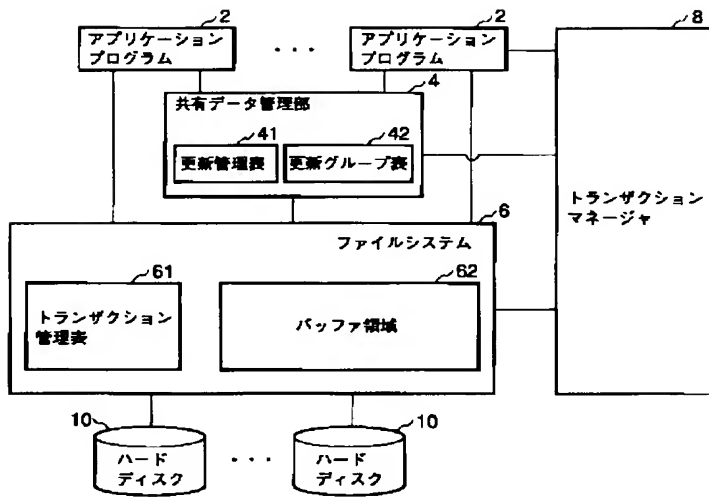
【図 26】

```

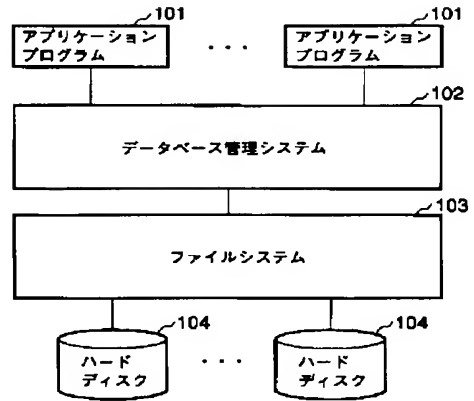
[FILE004]
<KVlist>
  <entry>
    <key>THYME</key>
    <value>200円</value>
    <less>
      <entry>
        <key>SESAME</key>
        <value>130円</value>
      </entry>
    </less>
    <greater>
      <entry>
        <key>VODKA</key>
        <value>1500円</value>
      </entry>
    </greater>
    <entry>
      <key>YOGHURT</key>
      <value>85円</value>
    </entry>
  </greater>
</entry>
</KVlist>

```

【図27】



【図43】



【図29】

```

[FILE003]
<KVlist>
  <entry>
    <key>MINT</key>
    <value>120円</value>
  </entry>
  <entry>
    <key>LEMON</key>
    <value>100円</value>
  </entry>
  <entry TXID="1" TXSTATE="UPDATE">
    <key>ORANGE</key>
    <value>250円</value>
    <oldvalue>210円</oldvalue>
  </entry>
  <entry>
    <key>PINEAPPLE</key>
    <value>980円</value>
  </entry>
</KVlist>

```

【図30】

```

[FILE004]
<KVlist>
  <entry>
    <key>THYME</key>
    <value>200円</value>
  </entry>
  <entry TXID="1" TXSTATE="UPDATE">
    <key>SESAME</key>
    <value>150円</value>
    <oldvalue>130円</oldvalue>
  </entry>
  <entry TXID="2" TXSTATE="REMOVE">
    <key>VODKA</key>
    <value>1500円</value>
  </entry>
  <entry>
    <key>YOGHURT</key>
    <value>85円</value>
  </entry>
</KVlist>

```

【図 3 1】

```

[FILE003]
<KVlist>
  <entry>
    <key>MINT</key>
    <value>120円</value>
    <less>
      <entry>
        <key>LEMON</key>
        <value>100円</value>
      </entry>
    </less>
    <greater>
      <entry>
        <key>ORANGE</key>
        <value>250円</value>
      </entry>
    </greater>
    <entry>
      <key>PINEAPPLE</key>
      <value>880円</value>
    </entry>
  </entry>
</KVlist>

```

【図 3 2】

```

[FILE004]
<KVlist>
  <entry>
    <key>THYME</key>
    <value>200円</value>
    <less>
      <entry>
        <key>SESAME</key>
        <value>150円</value>
      </entry>
    </less>
    <greater>
      <entry TXID="2" TXSTATE="REMOVE">
        <key>VODKA</key>
        <value>1500円</value>
      </entry>
    </greater>
    <entry>
      <key>YOGHURT</key>
      <value>85円</value>
    </entry>
  </entry>
</KVlist>

```

【図 3 3】

```

[FILE004]
<KVlist>
  <entry>
    <key>THYME</key>
    <value>200円</value>
    <less>
      <entry>
        <key>SESAME</key>
        <value>150円</value>
      </entry>
    </less>
    <greater>
      <entry>
        <key>VODKA</key>
        <value>1500円</value>
      </entry>
    </greater>
    <entry>
      <key>YOGHURT</key>
      <value>85円</value>
    </entry>
  </entry>
</KVlist>

```

【図 3 4】

```

[FILE001]
<KVlist>
  <entry>
    <key>KIWIFRUIT</key>
    <value>300円</value>
    <less>
      <entry TXID="3" TXSTATE="UPDATE">
        <key>DODO</key>
        <value>5000000円</value>
      </entry>
    </less>
    <link>FILE002</link>
    <less>
      <greater>
        <link>FILE005</link>
      </greater>
    </less>
    <greater>
      <entry>
        <key>ROSEMARY</key>
        <value>180円</value>
      </entry>
    </greater>
    <link>FILE003</link>
    <less>
      <greater>
        <link>FILE004</link>
      </greater>
    </less>
  </entry>
</KVlist>

```



【図35】

```

[FILE002]
<KVlist>
  <entry>
    <key>BISCUIT</key>
    <value>250円</value>
  </entry>
  <less>
    <entry>
      <key>APPLE</key>
      <value>400円</value>
    </entry>
  </less>
  <greater>
    <entry>
      <key>CHOCOLATE</key>
      <value>200円</value>
    </entry>
  </greater>
</KVlist>

[FILE003]
<KVlist>
  <entry>
    <key>MINT</key>
    <value>120円</value>
  </entry>
  <less>
    <entry>
      <key>LEMON</key>
      <value>100円</value>
    </entry>
  </less>
  <greater>
    <entry>
      <key>ORANGE</key>
      <value>250円</value>
    </entry>
    <greater>
      <entry>
        <key>PINEAPPLE</key>
        <value>880円</value>
      </entry>
    </greater>
  </greater>
</KVlist>

```

【図36】

```

[FILE004]
<KVlist>
  <entry>
    <key>THYME</key>
    <value>200円</value>
  </entry>
  <less>
    <entry>
      <key>SESAME</key>
      <value>150円</value>
    </entry>
  </less>
  <greater>
    <entry>
      <key>VODKA</key>
      <value>1500円</value>
    </entry>
    <greater>
      <entry>
        <key>YOGHURT</key>
        <value>85円</value>
      </entry>
    </greater>
  </greater>
</KVlist>

[FILE005]
<KVlist>
  <entry TXID="3" TXSTATE="UPDATE">
    <key>HOPOE</key>
    <value>880000円</value>
  </entry>
  <less>
    <entry>
      <key>GRAPE</key>
      <value>800円</value>
    </entry>
  </less>
  <greater>
    <entry>
      <key>ICECREAM</key>
      <value>450円</value>
    </entry>
  </greater>
</KVlist>

```

【図39】

```

<book>
  <chapter>
    <title>東京</title>
    <author>もんじゃ焼き</author>
  </chapter>
  <chapter>
    <title>名古屋</title>
    <author>海老フライ</author>
  </chapter>
  <chapter>
    <title>大阪</title>
    <author>たこ焼き</author>
  </chapter>
</book>

```

【図40】

```

<book>
  <chapter>
    <title>東京</title>
    <author>もんじゃ焼き</author>
  </chapter>
  <chapter>
    <title>名古屋</title>
    <author TXID="1" TXSTATE="REMOVE">海老フライ</author>
    <author TXID="1" TXSTATE="ADD">味噌かつ</author>
  </chapter>
  <chapter>
    <title>大阪</title>
    <author>たこ焼き</author>
  </chapter>
  <chapter TXID="2" TXSTATE="ADD">
    <title>博多</title>
    <author>からし明太子</author>
  </chapter>
</book>

```

【図37】

```

[FILE001]
<KVlist>
  <entry>
    <key>KIWIFRUIT</key>
    <value>300円</value>
  </entry>
  <less>
    <entry>
      <key>DODO</key>
      <value>500000円</value>
    </entry>
  </less>
  <link>FILE002</link>
</less>
<greater>
  <link>FILE005</link>
</greater>
</less>
<greater>
  <entry>
    <key>ROSEMARY</key>
    <value>180円</value>
  </entry>
</less>
  <link>FILE003</link>
</less>
  <greater>
    <link>FILE004</link>
  </greater>
</greater>
</entry>
</KVlist>

```

【図41】

```

<book>
  <chapter>
    <title>東京</title>
    <author>もんじゃ焼き</author>
  </chapter>
  <chapter>
    <title>名古屋</title>
    <author>味噌かつ</author>
  </chapter>
  <chapter>
    <title>大阪</title>
    <author>たこ焼き</author>
  </chapter>
  <chapter TXID="2" TXSTATE="ADD">
    <title>博多</title>
    <author>からし明太子</author>
  </chapter>
</book>

```

【図38】

```

[FILE002]
<KVlist>
  <entry>
    <key>BISCUIT</key>
    <value>250円</value>
  </entry>
  <less>
    <entry>
      <key>APPLE</key>
      <value>400円</value>
    </entry>
  </less>
  <greater>
    <entry>
      <key>CHOCOLATE</key>
      <value>200円</value>
    </entry>
  </greater>
</entry>
</KVlist>

[FILE005]
<KVlist>
  <entry>
    <key>HOOPOE</key>
    <value>980000円</value>
  </entry>
  <less>
    <entry>
      <key>GRAPE</key>
      <value>600円</value>
    </entry>
  </less>
  <greater>
    <entry>
      <key>ICECREAM</key>
      <value>450円</value>
    </entry>
  </greater>
</entry>
</KVlist>

```

【図42】

```

<book>
  <chapter>
    <title>東京</title>
    <author>もんじゃ焼き</author>
  </chapter>
  <chapter>
    <title>名古屋</title>
    <author>味噌かつ</author>
  </chapter>
  <chapter>
    <title>大阪</title>
    <author>たこ焼き</author>
  </chapter>
  <chapter>
    <title>博多</title>
    <author>からし明太子</author>
  </chapter>
</book>

```

フロントページの続き

(72)発明者 岐津 俊樹  
神奈川県川崎市幸区小向東芝町 1 番地 株  
式会社東芝研究開発センター内

(72)発明者 前田 誠司  
神奈川県川崎市幸区小向東芝町 1 番地 株  
式会社東芝研究開発センター内

(72)発明者 矢尾 浩  
神奈川県川崎市幸区小向東芝町 1 番地 株  
式会社東芝研究開発センター内

(72)発明者 矢野 浩邦  
神奈川県川崎市幸区小向東芝町 1 番地 株  
式会社東芝研究開発センター内

F ターム(参考) 5B082 GB04